UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/204,971	12/03/1998	DAVID LARS EHNEBUSKE	AT9-98-267	9311

7590 02/26/2002

DUKE W YEE
CARSTENS YEE & CAHOON
P O BOX 802334
DALLAS, TX 75380

EXAMINER
INGBERG, TODD D

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 02/26/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

DETAILED ACTION

Claims 1 - 29 have been examined.

Claims 1 and 7 were amended.

Claims 1 - 29 remain rejected.

Information Disclosure Statement

From First Action On Merits

1. In the event the invention is related to the Assignee's (IBM) product FLOWMARK (IBM Trademark #2006543) the Examiner reminds Applicant of their duty to disclose. FLOWMARK dates back to filing for Trademark on December 16, 1993. Date of first use in commerce June 28, 1995.

2. The Applicant makes specific reference to Object Management Group (OMG), the Assignee (IBM) is a long time member of this standards organization for Object technology. In the event work of OMG is relevant to Object Request Broker (ORB a part of CORBA), application frameworks and Business Objects, the Applicant and Applicant's representative are reminded of their duty to disclose.

3. In the event the invention is related to the product of Newi from Integrated Objects - a joint venture of IBM and Softwright based in England, the Applicant and Applicant's representative are reminded of their duty to disclose.

Applicant's Response

"The Office Action questions whether the present invention is related to

FLOWMARK, a product of the present Assignee, Object Request Brokers (ORBs), and Newi, a product supposedly of a joint venture between the present assignee and Softwright. Applicants are not aware of any correlation between these products and the present invention. Applicants are under no obligation to provide information regarding references raised by the Examiner. If the Examiner believes that these products are relevant to the present application, the Examiner must cite references in a PTO Form 892. It is not Applicants duty to research references first raised by the Examiner. Applicants' only duty is to disclose references of which they are aware that are material to patentability."

Examiner's Response

The Examiner did ask if any of the following were related to the invention. Assignee's (IBM) product FLOWMARK (IBM Trademark #2006543) filing for Trademark on December 16, 1993. Date of first use in commerce June 28, 1995, Applicant own specific reference to Object Management Group (OMG) (***OMG is mentioned on page 3 of Applicant's own Specification*** and in the Martin reference on page 357 where IBM and OMG are mentioned together (just one of several reference points in the Martin reference). The Assignee (IBM) is a long time member of this standards organization for Object technology. And the product of Newi from Integrated Objects - a joint venture of IBM and Softwright based in England. Which the Examiner believes was mentioned in the Martin reference. Applicant has mentioned the standards group OMG as does the Martin reference. The Martin reference is already on a PTO-

892. The Assignees's commercial product FLOWMARK is in patent literature, with this office action patents of the Assignee on FLOWMARK will be listed on the PTO-892.

Specification

4. The amendment to the Specification has been entered.

5. The new title has been entered

Specification

Objection to terms in the Specification

From First Action On Merits

6. The following were not discernable in the Specification. Clarification is required.

Applicant's Response

"Applicants however, disagree with the Examiner's apparent objection to the use of various statements in the specification. Specifically, the Examiner objects to the statement ."

Examiner's Response

The Examiner has provide interpretations to terms in the Specification. Where the Applicant has provided a definition different than the interpretation the definition is given further review in view of the Specification. Where the term is not defined in the response the Examiner's interpretations are deemed a finding of fact.

From First Action On Merits

"Page 6 - "Typically, these rules always have to be true except possibly during some specific rule free period such as during the middle of a business operation." The Specification does not support this sentence nor is it clear."

Applicant's Response

"Typically, these rules always have to be true except possibly during some specific rule free period such as during the middle of a business operation" on page 6 as not being supported by the specification. The Examiner's objection is unclear. It is not understood how a statement that is in the specification is allegedly not supported by the specification. However, in an effort to clarify this statement for the Examiner, Applicants offer that this statement means that normally, for integrity to be maintained in a system, the integrity rules must all have an outcome that is "true." The only time when this requirement may be relaxed is possibly during a "rule free" period. An example of a "rule free" period is during the middle of a business operation. The statement is supported by the specification. In fact, the statement supports itself."

Examiner's Response

Merely making a statement in the Specification does not mean the statement is clear and discernable. The Applicant's apparent believe that if it is in print it is understood and supported is not persuasive. The interpretation of the response is that a "rule free" period is "when code other than rules are running". This is part of the normal art and supported by the Martin reference. The Martin reference teaches object oriented programming and rules.

From First Action On Merits

"Page 10 - "Point of Variability" - This term is not clearly defined in the Specification and the Examiner is unable to provide an interpretation because it is unsupported."

Applicant's Response

"The Examiner also objects to the use of the term "point of variability" on page 10 alleging that this term is not clearly defined. Applicants respectfully disagree. The next statement on page 10 clearly describes the point of variability as a point in an enterprise operation at which rules may be associated to implemented variable behavior. This may be done, for example, by associating different rules with the point of variability, by changing the process for rules already associated with the point, or the like. The specification is clear as to what this term means and the Examiner's allegation that the term is not clearly defined is without merit."

Examiner's Response

The Martin reference in the Chapter on Rules shows associating rules to points in the program. The Applicant's response is weak. The Examiner's interpretation is deemed a finding of fact.

From First Action On Merits

"Page 8 (and other pages) - Participant - This is not a term in the art and will be interpreted by the Examiner below."

Applicant's Response

"The Examiner also objects to the use of the term "participant" at page 8 of the specification as not being a term of art. The term "participant" as it is used in the present

application is clearly defined on page 11, lines 25-27 as being elements of a unit of work that are modified by the processing carried out in association with the unit of work. Thus, the term "participant" is not unclear."

Examiner's Response

The Applicant's definition of unit of work reads on a portion of a computer program such as a method/ rule which is taught by the Martin reference.

From First Action On Merits

"Page 11 - "Natural structuring is provided whether or not each unit of work is totally or partially automated." "

Applicant's Response

"The Examiner also objects to the use of the statement "Natural structuring is provide whether or not each unit of work is totally or partially automated," on page 11 although the Office Action does not state why the Examiner objects to this statement. However, for the Examiner's understanding, as stated two sentences prior to the one in question, a unit of work may follow the structures of work done by employees and associates of businesses. The statement in question merely states that natural structuring, in other words structuring as if the work were done by a employees and associates of businesses, is used regardless of whether the unit of work is totally or partially automated. Therefore, this statement is not unclear."

Examiner's Response

The response of the Applicant reads on the structuring of code for methods/rules. The Martin reference teaches implementation of "units of work" performed by employees and associates of business in the chapter on Rules and throughout the reference when taken as a whole.

From First Action On Merits

"Page 12 - "disappear" - This is not a normal term in the art. Interpreted as a local variable in function that is no longer kept when the routine is completed. Because the variables are not global. Scope Rules typically explain this concept."

Applicant's Response

"The Examiner further objects to the term "disappear" on page 12 as not being a "normal term in the art." The term "disappear" as it is used in the context of page 12 means, as the Examiner recognized, that the changes are not kept. Thus, the term is not unclear."

Examiner's Response

The response is viewed as an admission that the Examiner's interpretation is correct. Furthermore, the Examiner mentioned "scope rules" a term common in the art. The Applicant's response supports the Examiner's interpretation.

From First Action On Merits

"Page 12 - "visible to all subsequently started unit of work." Scope rules as per above."

Applicant's Response

"Lastly, the Examiner objects to the statement "visible to all subsequently started

unit of work" on page 12. The objection to this statement reads "Scope rules as per above" which does not make any sense. Applicants do not understand what the Examiner is attempting to say here. However, in an effort to clarify the statement objected to by the Examiner, this statement means that all subsequent units of work will be made aware of the change in state. Thus, this statement is not unclear. "

Examiner's Response

Applicant does not understand the term "scope rules". It is possible that the Application being on Rules in an Object Oriented system the term rules is clouded. In programming in general scope rules means if the contents of a variable is able to be accessed and if the contents is saved after the operation is complete. Stepping back from the term RULES in the context of the application the term "scope rules" to one of very ordinary skill in the art of programming should recognize the term. This issue is related to the term "disappear" above.

The clarification made by the Applicant is related to the term the Applicant did not respond to ***"State of the enterprise application*** - State of an object. The state of the enterprise application in an object oriented system.". The interpretation is that since each object has a state the scope rules for the object make changes to the object visible to other "units of work" (methods/rules). This too is not outside normal use of object oriented programming. State changes are supported by the Martin reference and specific rules for verifying the integrity of a state change are present in Chapter 10 on Rules.

From First Action On Merits

"State of the enterprise application - State of an object. The state of the enterprise application in an object oriented system."

Applicant's Response

Applicant did not provide a challenge to the Examiner's interpretation for this term.

Examiner's Response

The Examiner holds the interpretation as a finding of fact.

Status of Specification Objections

Applicant's Response

"In view of the above, Applicants respectfully submit that the specification is not unclear and that the objections should be withdrawn. Accordingly, Applicants respectfully request withdrawal of the objection to the specification."

Examiner's Response

The Examiner does not see any major disagreements in the interpretation of the terms. The objection to the terms in the Specification is withdrawn in view of the finding of facts above.

Claim Rejections - 35 U.S.C. § 112

From First Action On Merits

7. While applicant may be his or her own lexicographer, a term in a claim may not be given a meaning repugnant to the usual meaning of that term. See *In re Hill*, 161 F.2d 367, 73 USPQ 482 (CCPA 1947). The Examiner will list terms and provide an interpretation as the meaning the Examiner gained from the Disclosure. The Examiner will provide an interpretation as to their usual meaning or a meaning that one of ordinary skill in the art. A patent is a right to exclude others for providing a teaching. Excessive lexicography or unclear terms pose a risk to the claim that a patent is an actual teaching an not a mere legal document.

Applicant's Response

"The Office Action states that the claims are rejected under 35 U.S.C. § 112 but fails to make any note of which claims are being rejected or the basis upon which they are being rejected Thus, the Examiner has not satisfied the burden of establishing a rejection under 35 U.S.C. § 112. As a result, Applicants are under no obligation to provide any amendments or arguments to overcome this alleged rejection.

In this section of the Office Action, the Examiner sets forth a number of different "interpretations" of the terms used in the claims. The terms in the claims must be interpreted in light of the specification as one of ordinary skill in the art would interpret these terms. Thus, the Examiner's statements in this section with regard to his personal interpretation are not limiting to the present claims. However, in an effort to clarify the use of these terms to aid the Examiner,

Applicants offer the following example definitions. These definitions are not meant to imply any limitations to the use of these terms in the claim but are only offered as example definitions obtained from the current specification."

Examiner's Response

The Examiner did interpret the terms in view of the Specification and to the level of one of ordinary skill in the art. The Examiner sought clarity and the response has provided clarity. The meanings are considered a finding of fact.

From First Action On Merits

"Checking State Integrity (Application State) - It is old and well known that programs can have states. Analysis and Design of systems incorporate state chart diagrams to model the state a systems should be in at a particular time. The state chart are incorporated in programs that support multiple states. the ability to check the current state to determine operations is common and the test to ensure the current state is common. The Applicant is not claiming to have invented checking the state but a means of checking the state in a RULE prior to the conclusion of an operation that will write to persistent storage (commit) or will not write to persistent storage."

Applicant's Response

"Checking State Integrity" - checking to see if the states of the participants in the unit of work maintain some feature that is important to the integrity of the application state and based on the collective judgment of the rules, determining if the unit of work as a whole passes the integrity check (page 12, lines 17-23);

Examiner's Response

The terms of visible (scope rules) and the fact that objects have states and the integrity rule Martin provides meets the definition of Checking State Integrity.

From First Action On Merits

"Externalized Rules - The Applicant's disclosure speaks of an external database containing rules. The concept of storing rules in a single location is not a new concept and one of ordinary skill in the art would utilize a static object. The Examiner interprets the term to mean more than merely storing rules in a single location. The Examiner interprets the term to mean more. In addition to storing in a single location, Externalized rules are not embedded in code requiring a programmer to maintain. They are linked to the executable code but are separate and able to be maintained by a user with domain knowledge. Domain knowledge being knowledge of business policy."

Applicant's Response

"Externalized Rules" - rules that are external to the applications implementing them (page 3, line 21 to page 4, line 7).

Examiner's Response

The Examiner's use of the Rule Editor and in Chapter 10 and the ability for "Rules can be attached to any diagram in the previous chapter", shows they are "external" to the applications implementing them. The Martin reference must be taken as a whole.

From First Action On Merits

"Unit of Work - The Specification states one of ordinary skill in the art will understand this term. However, the term must be understood in the contact in which it is used. " Correction contact should read context.

Applicant's Response

"Unit of Work" - representations of pieces of business work which define each business context in which they are carried out (page 11, lines 21-23; also an example is shown in Figure 3);

Examiner's Response

As covered prior a "unit of work" is met by a methods/ rule.

From First Action On Merits

"Committing it - In database art this term typically means to write to persistent storage. This interpretation is consistent with the description on page 12 of the Specification."

Applicant's Response

"Commit" - a process of determining whether to keep the changes in state made by the processing of a unit of work (page 12, lines 11-14; page 13, lines 19-23); also to keep the changes in state by storing them in persistent storage (page 12, lines 25-28);

Examiner's Response

Consistent with the interpretation given in the rejection.

From First Action On Merits

"Aborting it - The term to one of ordinary skill in the art means to terminate. However, terminate has many meanings as well. Such as terminate with an exception handle OR terminate the

method and return control with the use of a NULL (C programming language) etc. The Applicant has provided a strict limitation with the term abort. It appears it can meaning anything except write to persistent storage which is the ***commit*** as interpreted above."

Applicant's Response

"Abort" - to stop processing or to cause a process to not complete (page 12, lines 1-3);

Examiner's Response

Consistent with the Applicant's publicly held definition of abort. To stop and not to complete a process.

From First Action On Merits

"Participant - The Examiner interpreted this to mean methods of object(s). The term is critical to the application and further explanation is needed. The term in view of the figures is the suggested approach to describing this term."

Applicant's Response

"Participant" - see the definition of participant set forth above in the response to the objections to the specification;

Examiner's Response

Covered above as Applicant points out.

From First Action On Merits

"Disappear - the values in variables are temporary and not maintained after a program exits. This is common in local variables of a routine that exits. This is part of scope rules of variables in programming."

Applicant's Response

"Disappear" -- see the definition of disappear set forth above in the response to the objections to the specification;

Examiner's Response

Covered above as Applicant points out.

From First Action On Merits

"Visible to all subsequently started unit of work - Often visibility is often referred to in terms of scope. As in the ability to access methods and attributes. This does not seem to be the intended meaning of the term. "

Applicant's Response

"Visible to all subsequently started unit of work" - see the definition of this phrase set forth above in the response to the objections to the specification;

Examiner's Response

Covered above as Applicant points out.

Applicant's Response

"State of the enterprise application" - the state of the enterprise application as stated by the Examiner on page 6 of the Office Action.

Examiner's Response

No challenge to this term.

Status of Rejection 35 U.S.C § 112

Examiner's Response

The Examiner does not see any major disagreements in the interpretation of the terms. The objection to the terms in the Claim language the rejection is withdrawn in view of the finding of facts above.

Common Knowledge of Programming

From First Action On Merit

8. The following are considered to be common knowledge to one of ordinary skill in the art at the time of invention.

Applicant's Response

"In this section of the Office Action, the Examiner also provides some allegedly "common knowledge of programming" terms and definitions obtained from the Martin reference and the IBM Computer Dictionary. It is not clear whether the Examiner is asserting a rejection of the claims or not in this section. If the Examiner is stating a rejection, the Examiner is required to set forth which claims are being rejected and the basis for each rejection. Therefore, Applicants need not respond to the Examiner's statement of supposed "common knowledge of programming" terms.

In an order to expedite matters, however, Applicants do not disagree that these definitions are possible definitions of the terms. Regardless, the terms as used in the pending claims are defined by the specification regardless of the definitions chosen by the Examiner. Thus, Applicants' claims are not limited to the definitions obtained from the Martin reference or the IBM Computer Dictionary."

Examiner's Response

Terms and their interpretations are a part of the written record of the prosecution. The findings are presented during the prosecution. In the event, the Applicant disagreed with the meanings the Applicant can respond with a seasonable challenge. The terms and their meanings in the written record are considered a finding of fact.

Applicant's Response

"In this section of the Office Action, the Examiner also provides a subsection entitled "Terms of the Applicant's vs. Martin Reference." It is not clear what the Examiner is attempting to state in this section other than the Martin reference allegedly teaches "Integrity Rules." The Examiner has not stated how this alleged teaching affects the present application. Because the Examiner has not set forth a clear rejection in this section of the Office Action, Applicants are under no obligation to respond. However, it should be noted that Applicants are not bound by any statement set forth in this section of the Office Action and respectfully traverse any allegation with regard to any interpretation by the Examiner of terms or phrases used in the present

application that are not supported by the present specification. Again, the present application and the pending claims are not limited to any definitions found in the Martin reference."

Examiner's Response

Critical to the invention is the ability to check state integrity. The Examiner highlighted the feature in Martin, a feature that performs state integrity checking.

Status of Finding of Facts for Terms in the Art

The following are considered a finding of fact in this prosecution.

state - objects have states [**Martin**, page 143].

state transition diagrams - Modeling system behavior the use of state charts is to indicate the appropriate state given the current status of the environment [**Martin**, pages 104-5,114,158,160,164,165,206,216,235,285,293,376,398, and changes to state 397-8, 103,119,281,289, 113-4].

scope rules of variables - The ability to use a variable in portions of a program [**IBM** Dictionary, Scope, page 597].

commit - Typically a database term. The operation to write permanently to persistent storage. The Specification uses this term in this manner as well [**IBM** Dictionary page 119].

rollback - Also a database term. The ability to undo an operation such as a write prior to making it permanent [**IBM** Dictionary page 586].

abnormal end of task (abend) - Termination of a task before its completion because of an error condition that cannot be resolved by recovery facility while the task is executing. [**IBM**, page 1].

Terms of the Applicant's vs. Martin Reference

9. To avoid confusion the Examiner points out the Martin reference uses the terms Integrity Rule. Martin's use of the term on page 137 says it is used to make sure something must be true. This could be used to verify the state. If the state the program is suppose to be in is compared to the actual state the integrity rule must be true or the THEN condition in the rule is executed. The THEN would be the non commit and TRUE condition is the commit. Also in the same section of Martin the ability to perform a precondition rule and operation post condition rule exist.

Martin is more explicit about checking the actual state of an object on page 143 with RULES.

Claim Rejections - 35 U.S.C. § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1 - 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Martin** in view of Common Knowledge of Programming as taught by **Code Complete** and definitions as provided by the **IBM** Computer dictionary.

Claim 1

Martin teaches a method for performing general integrity checks using rules in an application running on a data processing system (**Martin**, page 133, 138 - 142, structure of Rules)

comprising: identifying a point in a unit of work where application state integrity is to be verified (**Martin**, page 143, object state rules), wherein the unit of work includes a plurality of participants (**Martin,** page 143, last paragraph, linked to appropriate items in OO diagram); obtaining rules associated with each participant in the unit of work (**Martin,** page 144, Box 10.3); and responsive to obtaining the rules (**Martin**, page 136, Rules Linked to Diagrams), running the rule obtained for each of the participants to verify the integrity of an application state (**Martin**, page 143, object state rules), according to the plurality of participants; general integrity checks running on a data (**Martin**, page 133, 138 - 142, structure of Rules). Martin does not explicitly teach the programming constructs of responsive to determining that the unit of work is to be completed. Although, Martin clearly supports Rules and the testing for conditions the reference does not explicitly state RULES can be used to determine if a write operation should be performed or not. The following overviews the teaching of **Martin**.

Martin Reference teaches RULES

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

What **Martin** does give is the endless possibilities of the conditions and the endless possibilities of the operations, such as if a state condition is wrong don't write (negative results) if the state operation is correct (positive results) perform a write operation. Examiner, holds determinations when to perform a write operation and when not to perform a write are issues of normal use and considered part of being a artisan of ordinary skill in the art. The term for employing common sense in programming is called Defensive Programming. It is the reference **"Code Complete"** by Steve Mc Connell that teaches defensive programming like on page 97 of **Code Complete** "Garbage In Does Not Mean Garbage Out". In other words test before you use data. Therefore, it would have been obvious to one of ordinary skill at the time of invention to use the teaching of Martin's RULES to perform defensive programming, because " ... it's the recognition that programs will have problems and modifications, and that a smart programmer will develop code accordingly". (**Code Complete**, page 94, 5.6 Defensive Programming, Key Point)

Claim 2

The method of claim 1, further comprising: responsive to a negative result obtained by running the rules (as per claim 1) , aborting the unit of work (**IBM** page 1, abnormal termination).

Claim 3

The method of claim 1, further comprising: responsive to a positive result obtained by running the rules (as per claim 1), committing the unit of work (**IBM**, page 119, make changes permanent as in write).

Claim 4

The method of claim 1, wherein each participant is associated with a name and wherein the step of obtaining rules associated with each participant in the unit of work comprises obtaining rules based on the name associated with the participant (**Martin,** page 136, Rules Linked to Diagrams).

Claim 5

The method of claim 4, wherein the plurality of participants are a plurality of objects and wherein the name associated with an object within the plurality of objects is the class name of a participating object (**Martin,** page 144, Box 10.3, "Customer").

Claim 6

The method of claim 1, wherein each participant is associated with a name (**Martin,** page 144) , wherein the unit of work is associated with a type (Object Technology - by definition each object has a type), and wherein the step of obtaining rules associated with each participant in the unit of work comprises obtaining rules based on the name associated with the participant and the type associated with the unit of work (**Martin,** page 136, Rules Linked to Diagrams and pages 138 - 140).

Claim 7

The method of claim 1, wherein at least zero integrity checking rules are associated with each participant within the plurality of participants (**Martin,** page 140, Stimulus ON or OFF).

Claim 8

Martin teaches a method in a data processing system for performing general integrity checks using rules (**Martin,** 143, object-state rule), the method comprising: detecting a commit for a unit of work (**Martin**, 140 - 141, ability to have conditionals); identifying participants in the unit of work in response to detecting the commit for the unit of work (**Martin**, 140 - 141, ability to have conditionals) ; determining whether rules are present for the participants in the unit of work (**Martin**, 140 - 141, ability to have conditionals); running the rules for participants identified as having at least one rule (**Martin,** page 140, Stimulus ON or OFF) , determining whether a violation of an integrity rule within the rules identified for any participant has occurred (**Martin**, 140 - 141, ability to have conditionals); and committing the unit of-work depending on the results of running the rules (**Martin**, 140 - 141, ability to have conditionals). What Martin does not explicitly teach is the basics of programming. Programmers of ordinary skill in the art know to test conditions to determine what operations should occur. Test of weather to write or not to write are common programming constructs. The Rules taught in Martin provide a RULES framework to support old and well known operations such as to write or not to write based on the results of testing a RULE. The Martin reference shows the anatomy of a RULE to look like the following:

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

The operation to write is performed if the condition is true after the integrity check. If the condition to not write is present the operation to not write is performed. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing write operations and aborting normal operations, because it makes programs reliable.

Claim 9

The method of claim 8 further comprising: aborting completion of processing by the unit of work in response to a determination that a violation of a rule has occurred; and committing completion of processing by the unit of work in response to a determination that no violation of a rule has occurred as per claim 8.

Claim 10

The method of claim 8, wherein each participant has zero or more rules associated therewith (**Martin,** page 140, Stimulus ON or OFF).

Claim 11

The method of claim 8, wherein each rule associated with a unit of work has available for use each participant within the unit of work (as per claim 8 the structure of a rule).

Claim 12

Martin teaches an enterprise application for use in a computer (**Martin,** RULES, chapters 4, 5 and 10), the enterprise application comprising: a unit of work (**Martin**, page 61, Objects), wherein the unit of work accumulates participants that affect a state of the enterprise application (**Martin**, page 61, Object Behavior Analysis - States); a plurality of business rules (**Martin,** page 133, Rules), wherein the plurality of rules are used to verify the integrity of the application state (**Martin,** page 143); and a unit of work control point, wherein the unit of work control point locates applicable rules for participants in response to an activation of the unit of work (**Martin,** pages 140 - 142, diagram of rules structure) to complete processing-of the unit of work. What Martin does not explicitly teach is the operations to be conducted in the RULES. Martin is a reusable construct for programmers to use as the problem solution warrants. An anatomy of Martin's RULE is :

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

Martin does not explicitly teach how to perform common operations such as to write or not to write based on the result of the integrity check. Martin provides the means for testing the state and based on the result to perform operations. It is well known to one of ordinary skill in the

art how to perform a write operation. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing write operations (commit) and aborting normal operations, because it makes programs reliable.

Claim 13

The enterprise application of claim 12, wherein the activation of the unit of work control point for the unit of work (Martin, page 140 - 142) is initiated by a commit instruction to the unit of work as per claim 12.

Claim 14

The enterprise application of claim 12, wherein the control point identifies applicable rules for all of the participants in the work of unit (**Martin**, 140 - 141, ability to have conditionals).

Claim 15

The enterprise application of claim 12, wherein the control point applies applicable rules to a portion of the participants in the work of unit (**Martin**, 140 - 141, ability to have conditionals).

Claim 16

The enterprise application of claim 12, wherein the applicable rules are identified based on a name associated with the participant (**Martin,** page 144).

Claim 17

The enterprise application of claim 12, the participant is an object and wherein the name is the class name of the participating object (**Martin,** page 144).

Claim 18

The enterprise application of claim 17, wherein the unit of work is associated with a type and wherein the applicable rules also are identified based on the type associated with the unit of work (**Martin,** page 144).

Claim 19

Martin teaches a data processing system for performing general integrity checks using rules (**Martin,** page 143) in an application running on a data processing system (**Martin,** page 143) comprising: identifying means for identifying a point in a unit of work where application state integrity is to be verified (**Martin**, page 133, 138 - 142, structure of Rules and page 143), wherein the unit of work includes a plurality of participants (**Martin**, page 133, 138 - 142, structure of Rules), first obtaining means, responsive to determining that the unit of work is to be completed (**Martin**, page 133, 138 - 142, structure of Rules), for obtaining rules associated with each participant in the unit of work (**Martin**, page 133, 138 - 142, structure of Rules); and second obtaining means, responsive to obtaining the rules (**Martin**, page 133, 138 - 142, structure of Rules), for running the rules obtained for each of the participants to verify the integrity of the system (**Martin**, page 143), according to the plurality of participants (**Martin**, page 133, 138 - 142, structure of Rules). What Martin does not explicitly teach is the operations to be conducted in the RULES. Martin is a reusable construct for programmers to use as the problem solution warrants. An anatomy of Martin's RULE is :

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

Martin does not explicitly teach how to perform common operations such as termination/completion of an operation. It is well known to one of ordinary skill in the art how to terminate an operation. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing simple operations like terminating an operation, because it makes programs reliable.

Claim 20

The data processing system of claim 19, further comprising: aborting means, responsive to a negative result obtained by running the rules (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule), for aborting the unit of work (**IBM** page 1, abnormal termination).

Claim 21

The data processing system of claim 19, further comprising: committing means, responsive to a positive result obtained by running the rules (**Martin**, page 133, 138 - 142, structure of Rules and

page 143 state integrity rule), for committing the unit of work **(IBM**, page 119, make changes permanent as in write).

Claim 22

Martin teaches a data processing system for performing general integrity checks using rules **(Martin**, page 143), the data processing system comprising: detecting-means for detecting a commit for a unit of work (**Martin**, page 133, 138 - 142, structure of Rules) ; identifying means for identifying participants in the unit of work in response to detecting (**Martin**, page 133, 138 - 142, structure of Rules) the commit for the unit of work; first determining means for determining whether rules are present for the participants in the unit of work;(**Martin,** page 140, Stimulus ON or OFF) running means for running the rules for participant identified as having at least one rule (**Martin,** page 140, Stimulus ON or OFF) ; second determining means for determining whether a violation of an integrity rule within the rules identified for any participant has occurred (**Martin**, page 133, 138 - 142, structure of Rules) ; and committing means for committing the unit of work depending on the results of running the rules (**Martin**, page 133, 138 - 142, structure of Rules). What Martin does not explicitly teach is the operations to be conducted in the RULES. Martin is a reusable construct for programmers to use as the problem solution warrants. An anatomy of Martin's RULE is :

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

Martin does not explicitly teach how to perform common operations such as to write or not to write based on the result of the integrity check. Martin provides the means for testing the state and based on the result to perform operations. It is well known to one of ordinary skill in the art how to perform a write operation. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing write operations (commit) and aborting normal operations, because it makes programs reliable.

Claim 23

The data processing system of claim 22 further comprising: aborting means for aborting completion of processing by the unit of work in response to a determination that a violation of a rule has occurred; and committing means for committing completion of processing by the unit of work in response to a determination that no violation of a rule has occurred as per claim 22.

Claim 24

The data processing system of claim 22, wherein each participant has zero or more rules associated therewith (**Martin,** page 140, Stimulus ON or OFF).

Claim 25

Martin teaches a computer program product for performing general integrity checks using rules in an application running on a computer program product (**Martin,** page 143, integrity check for object state), comprising: first instructions for identifying a point in a unit of work where application state integrity is to be verified (as per above), wherein the unit of work includes a plurality of participant (**Martin**, pages 138 - 140), second instructions for responsive to determining that the unit of work is to be completed (**Martin**, page 133, 138 - 142, structure of Rules), obtaining rules associated with each participant in the unit of work; and third instructions for responsive to obtaining the rules (**Martin**, page 133, 138 - 142, structure of Rules), running the rules obtained for each of the participants to verifying the integrity of the system (**Martin,** page 143, integrity check for object state), according to the plurality of participants (**Martin**, pages 138 - 140). What Martin does not explicitly teach is the operations to be conducted in the RULES. Martin is a reusable construct for programmers to use as the problem solution warrants. An anatomy of Martin's RULE is :

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages)

Martin does not explicitly teach how to perform common operations such as termination of an operation. It is well known to one of ordinary skill in the art how to terminate an operation. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing simple operations like terminating an operation, because it makes programs reliable.

Claim 26

The computer program product of claim 25, further comprising: first instructions for responsive to a negative result obtained by running the rules (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule), aborting the unit of work (**IBM** page 1, abnormal termination).

Claim 27

The method of claim 25, further comprising: first instructions for responsive to a positive result obtained by running the rules (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule), committing the unit of work (**IBM**, page 119, make changes permanent as in write).

Claim 28

Martin teaches a computer program product in a data processing system for performing general integrity checks using rules (**Martin**, page 142, state integrity rule) , the computer program product comprising: first instructions for detecting (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule) a commit for a unit of work; second instructions for

identifying participants in the unit of work in response to detecting (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule) the commit for the unit of work; third instructions for determining whether rules are present (**Martin,** page 140, Stimulus ON or OFF) for the, participants in the unit of work; fourth instructions for running the rules for participants identified as having at least one rule; fifth instructions for determining (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule) whether a violation of an integrity rule within the rules identified for any participant has occurred; and sixth instructions for committing the unit of work depending on the results, of running the rules (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule). What Martin does not explicitly teach is the operations to be conducted in the RULES. Martin is a reusable construct for programmers to use as the problem solution warrants. An anatomy of Martin's RULE is :

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(**Note:** the construct closely resembles the structure of a CASE/switch statement in many languages) Martin does not explicitly teach how to perform common operations such as to write or not to write based on the result of the integrity check. Martin provides the means for testing the state and based on the result to perform operations. It is well known to one of ordinary skill in the art

how to perform a write operation. Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of Martin with common knowledge of programmers on performing write operations (commit) and aborting normal operations, because it makes programs reliable.

Claim 29

The computer program product of claim 28 further comprising: first instructions for aborting completion (**IBM** page 1, abnormal termination) of processing by the unit of work in response to a determination that a violation of a rule (**Martin**, page 133, 138 - 142, structure of Rules and page 143 state integrity rule) has occurred (**IBM** page 1, abnormal termination); and second instructions for committing completion of processing by the unit of work in response to a determination that no violation of a rule has occurred (**IBM**, page 119, make changes permanent as in write).

Response to Arguments

12. Applicant's arguments filed October 9, 2001 have been fully considered but they are not persuasive.

Arguments related to grounds of Rejection under 35 U.S.C. § 103, Obviousness

From First Action On Merits

"The Office Action rejects claims 1-29 under 35 U.S.C. § 103(a) over Martin, Principles of Object-Oriented Analysis and Design, 1993 in view of alleged common knowledge of

programming as taught by Complete Code and definitions as provided by the IBM Computer Dictionary. This rejection is respectfully traversed.

With regard to claim 1, the Office Action states:

Martin teaches a method for performing general integrity checks using rules in an application running on a data processing system (Martin, page 133, 138 - 142, structure of Rules) comprising: identifying a point in a unit of work where application state integrity is to be verified (Martin, page 143, object state rules), wherein the unit of work includes a plurality of participants (Martin, page 143, last paragraph, linked to appropriate items in 00 diagram); obtaining rules associated with each participant in the unit of work (Martin, page 144, Box 10.3); responsive to obtaining the rules (Martin, page 136, Rules Linked to Diagrams), running the rule obtained for each of the participants to verify the integrity of an application state (Martin, page 143, object state rules), according to the plurality of participants; general integrity checks running on a data (Martin, page 133, 138 - 142, structure of Rules). Martin does not explicitly teach the programming constructs of responsive to determining that the unit of work is to be completed. Although, Martin clearly supports Rules and the testing for conditions the reference does not explicitly states RULES can be used to determine if a write operation should be performed or not. The following overviews the teaching of Martin.

Martin Reference teaches RULES

Testing the Integrity of an Object (page 143)

When condition (Chapter 10 a format for Rules logic structure testing the state of the Object)

Perform operation

When condition

Perform operation

(Note: the construct closely resembles the structure of a CASE/switch statement in many languages)"

Applicant's Response

What Martin does give is the endless possibilities of the conditions and the endless possibilities of the operations, such as if a state condition is wrong don't write (negative results) if the state operation is correct (positive results) perform a write operation. Examiner, holds determinations when to perform a write operation and when not to perform a write are issues of normal use and considered part of being a artisan of ordinary skill in the art. The term for employing common sense in programming is called Defensive Programming. It is the reference "Code Complete" by Steve McConnell that teaches defensive programming like on page 97 of Code Complete "Garbage In Does Not Mean Garbage Out". In other words test before you use data. Therefore, it would have been obvious to one of ordinary skill at the tune of invention to use the teaching of Martin's RULES to perform defensive programming, because ".. .it's the recognition that programs will have problems and modifications, and that a smart programmer will develop code accordingly". (Code Complete, page 94, 5.6 Defensive Programming, Key Point).

Examiner's Response

The description of the rejection is correct. The references teach the basic tools of the trade available to an artisan of ordinary skill in the art.

Applicant's Response

"Claim 1, which is representative of claims 19 and 25 with regard to similarly recited subject matter, reads as follows:

1. A method for performing general integrity checks using rules in an application running on a data processing system comprising:

identifying a point in a unit of work where application state integrity is to be verified, wherein the unit of work includes a plurality of participants; responsive to determining that the unit of work is to be completed, obtaining rules associated with each participant in the unit of work; and responsive to obtaining the rules, running the rules obtained for each of the participants to verify the integrity of an application state, according to the plurality of participants.

It should first be noted that the Martin reference is a reference comprising over 400 pages of text. Therefore, Applicants in this Response are only addressing those sections explicitly cited by the Office Action."

Examiner's Response

The references must be taken as a whole.

Applicant's Response

"Martin is a general text book that includes a section on the use of rules in object oriented programming. While Martin generally teaches the use of rules, and even integrity rules, Martin does not teach or even suggest the specific use of rules set forth in claim 1. A general teaching does not render every specific use of the general teaching obvious. The following are Applicants' responses to the Office Action's allegations regarding the specific features recited in claim 1. Martin does not teach or suggest any of these features."

Examiner's Response

The 1992 Martin publication provides a teaching on Object Oriented CASE tools. The Martin reference provides teachings on Rules and common programming constructs. The references teach well know programming constructs and tools available to one of ordinary skill in the art well prior to the time of invention.

Applicant's Response

"The Office Action alleges that Martin teaches a method of performing general integrity checks using an application running on a data processing system at pages 133 and 138-142. While it is true that Martin does teach the use of integrity rules and states that these rules indicate that something must be true (page 137), Martin does not teach the specific use of integrity rules set forth in claims 1, 19 and 25 (hereafter only referred to as claim 1), as discussed below.

The Office Action alleges that Martin teaches identifying a point in a unit of work where application state integrity is to be verified merely because Martin allegedly teaches object state rules at page 143. First, Martin does not teach or even suggest units of work as the term is used in

the present application. As set forth above, a unit of work is a piece of business work which defines each business context in which it is carried out. While Martin teaches that business policies may be represented as rules (see page 133), Martin makes not mention or even suggestion regarding units of work."

Examiner's Response

Based on Applicant's response the Martin reference does in fact teach a "unit of work". the response indicates that a method/rule meets the limitation. Furthermore, the rejection states that Martin does not ***explicitly*** teach unit of work. As in Martin does not use the same terms. However, Martin does teach unit of work in terms of methods/ rules. Applicant's argument is not persuasive.

Applicant's Response

"Because Martin does not teach units of work, Martin cannot be found to teach or even suggest identifying a point in a unit of work where application state integrity is to be verified. While Martin teaches object state rules on page 143, all that is stated is "Object state rules are identified in object-structure analysis. They are associated with diagrams, such as the data-structure diagram, object-relationship diagram, or composed-of diagram." It is not seen how such a general statement somehow teaches the very specific feature of identifying a point in a unit of work where application state integrity is to be verified, as recited in claim 1. The description of object state rules on page 143 of Martin does not make any mention of units of work, let alone identifying a point in a unit of work where state integrity is to be verified."

Examiner's Response

In view, of the unit of work response above the Martin reference does in fact have a Rule for integrity checking. The Martin reference does not provide the limits the Applicant appears to be imposing on the Martin reference. In fact, the Martin reference states the rules can be attached to any diagram (as in any point) and mentions the diagrams are executable. that is to say they represent code that models the problem domain. Applicant's argument is not persuasive.

Applicant's Response

"Furthermore, even if it were interpreted that Martin teaches units of work, Martin still does not teach identifying a point in a unit of work where application state integrity is to be verified. As noted above, the general statement that there are object state rules, that they are identified in object-structure analysis, and that they may be associated with diagrams, has nothing to do with identifying a point in a unit of work where application state integrity is to be verified. The Examiner is reading features into the Martin reference that is simply not there in order to arrive at Applicants' claimed invention having first had benefit of Applicants' disclosure. In other words, the Examiner is engaged in hindsight reconstruction in which the Examiner is conjuring teachings from the reference that simply are not there."

Examiner's Response

The combination of the two responses above cover the argument present. In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense

necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Applicant's Response

"The Office Action alleges that Martin teaches that a unit of work includes a plurality of participants merely because Martin allegedly mentions "three more rule windows that are linked to appropriate items in OO diagrams" (page 143, last paragraph). It is not understood how linking rule windows to object oriented diagrams teaches a unit of work having a plurality of participants. There is no correlation between the feature recited in claim 1 and the cited portion of Martin. Linking rule windows to object oriented diagrams has nothing to do with a unit of work having a plurality of participants."

Examiner's Response

The Martin reference when taken as a whole teaches that OO-CASE tools are for providing enterprise solutions (page 61). Enterprises having many participants in fact Chapter 6 mentions the categories of objects (Categorizing Objects). Chapter 7 teaches the interaction among objects and how they relate. Specifically, in the context of participants the Martin reference uses the foundation knowledge of the early chapters and Chapter 13 Responsibility Driven Design teach the use of "actors". These basics in view of the use of Rules meet the limitations when taking the reference as a whole. When looking at the citation in the rejection the

ability to link multiple rules of the enterprise in the rules editor meets the limitations. The limitation of "unit of work" was already covered.

Applicant's Response

"The Office Action further alleges that Martin teaches obtaining rules for each of the participants in the unit of work merely because Martin provides example of rules associated with diagrams of object oriented analysis in Box 10.3 on page 144. Box 10.3 of Martin does not teach a unit of work, let alone obtaining rules for each participant in the unit of work. The only thing that Box 10.3 teaches is examples of rules for different types of object oriented diagrams. There is no teaching or even suggestion in Box 10.3 of using units of work, and definitely no teaching or suggestion of obtaining rules for each participant in a unit of work. Again, the Examiner is engaging in hindsight reconstruction by reading in teachings to the Martin reference that simply are not there."

Examiner's Response

When taking the reference as a whole the "actors" are defined and rules associated with the tasks of the participants. On page 144 the Rule associated with transactions is related to updating the Salary. This rule is part of a particiapants responsibility. Applicant's argument seems to have failed to taken the Martin reference as a whole.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so

long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Applicant's Response

"In addition, the Office Action alleges that Martin teaches running the rules obtained for each of the participants to verify the integrity of an application state merely because Martin allegedly teaches object state rules on page 143. The general teaching of object state rules does not provide any teaching or suggestion regarding running rules obtained for participants in a unit of work to verify the integrity of an application state."

Examiner's Response

Martin provides the programming constructs to identify participants and rules. Martin specifically, teaches a rule for integrity checking. The Examiner disagrees participants are taught by the Martin reference. In fact, participants are objects which the integrity rule can be applied. Applicant's argument is not persuasive.

Applicant's Response

"The general teaching of object state rules makes no teaching or suggestion of units of work, rules for participants in a unit of work, obtaining and running rules for participants in a unit of work, or that there is any correlation between rules for participants in a unit of work and

an application state. There simply is no correspondence between the section of Martin cited by the Office Action and the features recited in claim 1."

Examiner's Response

Examiner believes the arguments prior cover these limitations.

Applicant's Response

"In summary, Martin has nothing to do with the invention recited in claim 1. While Martin provides a number of general teachings, they do not teach or suggest any of the features recited in claim 1. Furthermore, the Examiner appears to be stretching the interpretation of Martin in an attempt to reach Applicants' claimed invention beyond any reasonable interpretation of the reference. Such a stretch of interpretation is based on hindsight reconstruction using Applicants' own disclosure as a guide."

Examiner's Response

The references teach old and well known programming constructs. In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971). Applicant are informed that the courts have held that "A reference anticipates a claim if

it discloses the claimed invention such that a skilled artisan could take its teachings in combination with his own knowledge of the particular art and be in possession of the invention.". *In re Graves, 36 USPQ2d 1697 (Fed. Cir. 1995); In re Sase, 207 USPQ 107 (CCPA 1980); In re Samour, 197 USPQ 1 (CCPA 1978).* The Federal Circuit has also held that anticipation is not shown where the difference between the claims and the reference is minor and such as would suggest itself to a person of ordinary skill in the art. *Connell v. Sears, Roebuch & Co., 220 USPQ 193 (Fed. Cir. 1983), Structural Rubber Prod. Co. v. Park Rubber Co., 223 USPQ 1264 (Fed. Cir. 1984).*

Applicant's Response

"Applicants do agree with the Office Action that Martin does not teach obtaining rules for each of the participants in a unit of work in response to determining that a unit of work is to be completed, as recited in claim 1. This is because Martin does not teach a unit of work, participants in a unit of work, or even that a unit of work may be completed. However, Applicants disagree with the Office Action regarding the allegation that Martin provides "endless possibilities" of conditions and operations such as write operations and that when to perform a write operation and when not to perform a write operation somehow makes obvious this feature. The Office Action also cites Code Complete as teaching defensive programming which the Office Action somehow links with Martin to allegedly modify the Martin reference so that a determination that a unit of work is to be completed is somehow made obvious."

Examiner's Response

The Examiner and Applicant are in agreement that Martin does not provide "endless possibilities" of conditions and operations. In fact, there is a finite number of combinations. The Examiner stands corrected. Unfortunately this is not grounds for allowance. The defensive programming reference shows the basic common sense one of ordinary skill in the art should know. Put simply, don't write (commit) when the system is in a status where the operation could be harmful (abort). The Applicant seems to think the concept is novel. The Martin reference teaches the programming constructs in an object oriented CASE tool to perform the claimed invention. This is deemed obvious. In re Graves cited above supports grounds of obvious rejections if an artisan of ordinary skill would have the knowledge at the time of invention on how to implement the basics of the tools of the trade as taught by Martin. Furthermore, the Applicant has misworded the actual rejection. The rejection stated "explicitly" and the prior response above shows the Applicant's response for "unit of work" is taught by Martin. The argument is not persuasive.

Applicant's Response

"First, the Applicant cannot follow the Examiner's reasoning. It is not at all clear how a general teaching of conditions can be found to teach the specific feature set forth in claim 1 of a determination that a unit of work is to be completed. The Office Action states that the conditions of when to write and when not to write make this feature obvious. Applicants do not claim to be the first to invent determining if a unit of work is to be completed. Rather, Applicants claim obtaining rules for participants in the unit of work when it is determined that a unit of work is to

be completed. Thus, the Office Action's example of a write condition has no bearing on the presently claimed features."

Examiner's Response

The Applicant is arguing about basic programming constructs as taught by Martin. This argument is redundant in view of the responses above. The anatomy of a rule is demonstrated with common programming constructs. Applicant thinks this has no bearing Examiner disagrees.

Applicant's Response

"Second, it is not clear how a general teaching of "defensive programming" may somehow be combined with the general teachings provided in Martin to arrive at the specific method set forth in claim 1. The Code Complete reference does not provide any teachings that cure the deficiencies in Martin set forth above. In other words, the general teaching of "defensive programming" does not render obvious "identifying a point in a unit of work where application state integrity is to be verified, wherein the unit of work includes a plurality of participants; responsive to determining that the unit of work is to be completed, obtaining rules associated with each participant in the unit of work; and responsive to obtaining the rules, running the rules obtained for each of the participants to verify the integrity of an application state, according to the plurality of participants," as recited in claim 1."

Examiner's Response

The Examiner rejected the integrity check with the Martin reference on checking the integrity. The defensive programming reference covers the use of common sense in programming and good programming habits. The remainder of the arguments are redundant.

Applicant's Response

"Although not explicitly cited in the rejection of claim 1, the IBM Computer Dictionary provides no additional teaching regarding any of the features of claim 1. That is, the combination of the IBM Computer Dictionary with Martin and Code Complete does teach or suggest any of the features in claim 1. None of these references have anything to do with the presently claimed invention recited in claim 1. The Examiner has taken extremely general teachings in a text book and attempted to stretch these teachings beyond reasonable interpretation and read additional teachings into the references that are not there in an attempt to encompass the very specific method set forth in claim 1. Such a stretch of the teachings an reading in of additional teachings is not supported by the references and is based solely on hindsight reconstruction using Applicants' own disclosure as guide."

Examiner's Response

The IBM dictionary shows the terms are well established and the Applicants lexicon has not provided a meaning outside the reasonable interpretation of the Assignee's publicly held meaning of the terms. The Martin reference teaches the checking of integrity and the Code Complete reference teaches common sense is to be used. The IBM dictionary underscores the

industry use of the terms which took years to culminate into a denotation and published in a computer dictionary. Applicant's argument is not persuasive.

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Applicant's Response

"With regard to claims 8, 22 and 28, neither Martin, Code Complete, or the IBM Computer Dictionary teaches or suggests detecting a commit for a unit of work. As set forth above, a commit in this context is a process for determining whether to commit the state changes made by the unit of work to the participants in the unit of work. The Office Action alleges that the mere teaching of having conditional rules in Martin somehow is the same as detecting a commit for a unit of work. Applicants respectfully disagree. Yet again, the Examiner is reading in teachings to the Martin reference that are not there. The general teachings of conditional rules does not teach or suggest detecting a commit of a unit of work."

Examiner's Response

The Examiner's interpretation is the calling of a method or rule when it ends it is a detectable event. One of ordinary skill in the art should know that the calling of a function transfers control and the completion returns control. This should not be in question. Furthermore, the ability to check a return code in programming is grossly old and well known. The argument is not to the level of ordinary skill in the art. And was covered in the FAOM.

Applicant's Response

"As previously noted above, Martin does not teach a unit of work. Therefore, Martin cannot teach detecting a commit of a unit of work. Furthermore, Martin does not teach a commit of a unit of work. A general teaching of a condition rule does not provide any teaching or suggestion of a determination of whether to commit the state changes made by a unit of work to the participants in the unit of work, i.e. a commit of a unit of work."

Examiner's Response

This argument is repetitive and not persuasive.

Applicant's Response

"In addition, claims 8, 22 and 28 include features that are similar to some of the features in claims 1, 19 and 25. Therefore, Martin, Code Complete and the IBM Computer Dictionary do not provide any teaching or suggestion of these features in claims 8, 22 and 28, as discussed above.

With regard to claim 12, as previously noted, none of the references teach or suggest a unit of work, participants in a unit of work, or locating rules for the participants in response to activation of the unit of work to complete the unit of work."

Examiner's Response

This argument is repetitive and not persuasive.

Applicant's Response

"In view of the above, Applicants respectfully submit that neither Martin, Code Complete, or the IBM Computer Dictionary, either alone or in combination, teach or suggest the features recited in claims 1, 8, 12, 19, 22, 25, 28. At least by virtue of their dependency on claims 1, 8, 12, 19, 22, 25 and 28, respectively, none of the references either alone or in combination teach or suggest the features set forth in dependent claims 2-7, 9-11, 13-18, 23-24, 26-27 and 29. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-29 under 25 U.S.C. § 103(a)."

Examiner's Response

Applicant and Examiner disagree. The Martin reference was apparently not taken as a whole. And the references teach common programming constructs available to one of ordianry skill in the art weel before the time of invention.

Applicant's Response

"In addition, none of the references teach or suggest any of the specific features set forth in the dependent claims 2-7, 9-11, 13-18, 23-24, 26-27 and 29. The Office Action provides

specific rejections of each of the dependent claims based on the combination of references discussed above. Each of these rejections is based on the same flawed interpretation and reading in of teachings discussed above and therefore, are traversed for similar reasons as set forth above.

In addition, because the dependent claims build off of the features recited in their respective independent claims, the deficiencies of the cited references are likewise applicable to the additional features set forth in the dependent claims. For example, because none of the references teach or suggest a unit of work or running rules for participants in a unit of work, as recited in claim 1, none of the references can be found to teach "responsive to a negative result obtained by running the rules, aborting the unit of work," as recited in claim 2."

Examiner's Response

The Applicant is arguing common concepts and programming constructs in the art. So common that the basic meanings of abort and commit are in the Assignee's own computer dictionary. The unit of work argument was answered by the response from Applicant and supported by the rejection as not ***explicitly*** mentioned but taught by the teaching of methods/ rules. Applicant's argument's have been fully considered and are not persuasive. This action is made FINAL.

Conclusion

13. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Correspondence Information

14. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to **Todd Ingberg** whose telephone number is **(703) 305-9775.** The Examiner can normally be reached on Monday through Thursday from 6:30 a.m. to 5:00 p.m.

If attempts to reach the examiner by telephone are unsuccessful, the **Examiner's Supervisor**, Leo Picard can be reached at **(703)308-0538.** Any response to this office action should be mailed to: **Director of Patents and Trademarks Washington, D.C. 20231**, or **Hand-delivered** responses should be brought to **Crystal Park II, 2121 Crystal Drive Arlington, Virginia, (Receptionist located on the fourth floor),** or **faxed.** The following **fax numbers** apply:

After Final	**(703) 746 - 7238**
Official	**(703) 746 - 7239**
Non Official/ Draft	**(703) 746 -7240**

Todd Ingberg

December 26, 2001